Pre-Grasp Sliding Manipulation of Thin Objects Using Soft, Compliant, or Underactuated Hands

Kaiyu Hang, *Member, IEEE*, Andrew S. Morgan, *Student Member, IEEE*, and Aaron M. Dollar, *Senior Member, IEEE*

***Abstract*—We address the problem of pregrasp sliding manipulation, which is an essential skill when a thin object cannot be directly grasped from a flat surface. Leveraged on the passive reconfigurability of soft, compliant, or underactuated robotic hands, we formulate this problem as an integrated motion and grasp planning problem, and plan the manipulation directly in the robot configuration space. Rather than explicitly precomputing a pair of valid start and goal configurations, and then in a separate step planning a path to connect them, our planner actively samples start and goal robot configurations from configuration sampleable regions modeled from the geometries of the object and support surface. While randomly connecting the sampled start and goal configurations in pairs, the planner verifies whether any connected pair can achieve the task to finally confirm a solution. The proposed planner is implemented and evaluated both in simulation and on a real robot. Given the inherent compliance of the employed Yale T42 hand, we relax the motion constraints and show that the planning performance is significantly boosted. Moreover, we show that our planner outperforms two baseline planners, and that it can deal with objects and support surfaces of arbitrary geometries and sizes.**

***Index Terms*—Grasping, manipulation planning, motion and path planning.**

I. Introduction

GRASPING is one of the most fundamental abilities that enables a robot to physically interact with objects and use them for different tasks. Based on the geometrical or topological representations of the target object, grasp planning algorithms have been developed to generate grasp contacts, hand configurations, hand poses and grasp policies [1], [2].

In order to simplify the physics modeling and focus on the synthesis of high quality grasps, traditional approaches assume that the object is static before a grasp is achieved [3]–[6].

In certain scenarios, although the object is graspable, pre-grasp rotation is required to change the object's orientation, so as to allow the execution of desired tool-use grasps [7], [8]. In cases where an object is initially not graspable at all, it can be first slid or pushed to a goal region, in which the robot hand can reach a larger portion of the object surface to make contacts, before grasp planning is conducted [9], [10].

Non-prehensile manipulation such as side-pushing and top-sliding are two typical approaches for pre-grasp manipulation, especially when the object is initially not graspable. When planning for pushing motions, due to the complex physics modeling or the need of object classification [9], [11], it is usually computationally expensive or infeasible to generalize to novel objects. For top-sliding, which is desired if the target object is thin, motion planning and force control are challenging if a fully-actuated hand is used. This is because the hand has to always maintain stable contacts on the object to ensure that the object slides with the hand, while it cannot press too hard in order to avoid damage.

In this work, as illustrated in Fig. 1, leveraged on the passive reconfigurability of soft, compliant, or underactuated hands, we focus on the problem of pre-grasp top-sliding for relocating and grasping thin objects, which is often used for cards, coins, rulers, books, and other flat objects. As has been demonstrated in many designs, underactuated hands provide high degrees of adaptability, which enables the fingers to passively reconfigure when external forces are applied [12]–[14]. Many of these same benefits exist in "soft" or compliant hands, although with a lesser degree of reconfiguration [15]. For top-sliding, this benefits the manipulation planning in three aspects: 1) explicit force control is not required to maintain stable contacts, which can be ensured by

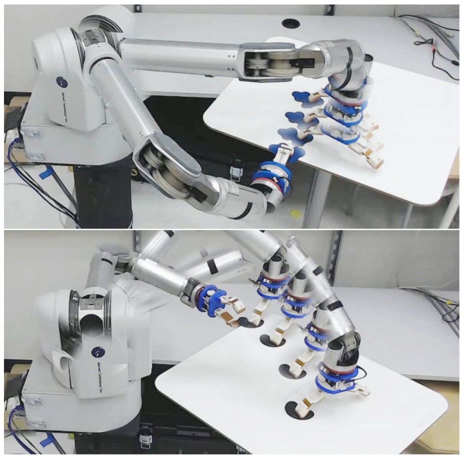
Fig. 1. A WAM arm installed with a Yale T42 hand grasps thin objects. The object is slid to an edge of the table and then grasped from the side.

Manuscript received September 10, 2018; accepted December 26, 2018. Date of publication January 11, 2019; date of current version January 25, 2019. This letter was recommended for publication by Associate Editor Y. Lou and Editor H. Ding upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Grant IIS-1734190. *(Corresponding author: Kaiyu Hang.)*

The authors are with the Department of Mechanical Engineering and Material Science, Yale University, New Haven, CT 06511 USA (e-mail: kaiyu.hang@yale.edu; andrew.morgan@yale.edu; aaron.dollar@yale.edu).

This letter has supplemental downloadable multimedia material available at http://ieeexplore.ieee.org, provided by the authors. The Supplementary Materials contain a video showing some key ideas of the paper and some robot execution records are included to make the proposed approach more intuitive and clear to the readers. This material is 5.41 MB in size.

Digital Object Identifier 10.1109/LRA.2019.2892591

a certain amount of finger reconfiguration; 2) the manipulation trajectory can violate the constraint manifold as allowed by the finger reconfiguration, and will not damage the hand or the object; and 3) some motion constraints can be relaxed to make the planning more flexible and efficient.

To this end, we formulate top-sliding manipulation as an integrated problem of motion and grasp planning, constrained on the manifold defined by the support surface. Rather than explicitly pre-computing the start and goal configurations [9], we provide a more efficient approach by modeling configuration sampleable regions for the planner to sample from, and then leave the planner to decide the start and goal autonomously. Briefly, the start region is modeled in terms of the object geometry to ensure valid initial contacts, which is maintained as a constraint throughout the entire manipulation. The goal region is modeled using the geometry of support surface's edges, around which thin objects can be potentially grasped. The planner actively samples start and goal configurations online, while randomly trying to connect a pair of them. Once connected, the path is considered a manipulation solution if the connected pair is verified to slide the object to a graspable pose. Finally, we extend the classical CBiRRT algorithm [16] by integrating configuration sampleable regions and pair validation to address motion and grasp planning in a unified framework.

We review related works in Section II and formalize the problem in Section III. Thereafter, the integrated motion and grasp planner will be detailed in Section IV. We evaluate our planner using a Yale T42 underactuated hand mounted on a WAM arm in Section V, and conclude our work in Section VI.

II. Related Work

Grasp planning has been an active research topic that involves a variety of subproblems ranging from, for example, contact-based grasp synthesis [2], [17], [18] and hand posture analysis [19], to grasp control [20], and learning of grasp policies [5]. Assuming a free-floating object, grasps can be planned neglecting environment constraints and motion feasibility is checked posteriorly [21], [22]. For objects on a tabletop, top-grasping policies are learned to establish grasp contacts on the object's side faces [5]. Furthermore, grasp and motion planning are integrated to ensure the motion feasibility [3], [4].

Since the traditional approaches assume the target object is static before grasping, they would fail in certain cases, such as when a thin object is on the tabletop and no collision-free grasp contacts can be found. Compliant mechanical designs can solve this problem by positioning and closing the hand above the object and utilizing contacts from the environment to pick up the object [14]. However, due to the uncertainties in modeling the dynamics in this process, the object's final pose in the grasp is unpredictable and makes it infeasible to ensure that the object will be grasped in a desired pose.

To this end, as a subset of nonprehensile manipulation skills, pre-grasp manipulation has been proposed. When the object is initially surrounded by other movable objects in clutter, pre-grasp object rearrangement is applied to make room for the hand to reach and grasp [23], [24]. When the object is distant, it can be slid to a closer area to allow for grasping [9]. To ensure the object is grasped in desired poses for future usage, the object is pre-manipulated to enable desired grasps [7], [8], [10]. In addition, toppling or tumbling actions can also be used to reconfigure the object to a better graspable pose [25].

However, it is not easy to generalize the approaches to novel objects when they are based on human demonstration, hand configuration adaptation, and object classification [7], [9]. For physical simulation based planning [10], [23], [25], it is computationally expensive, not robust to uncertainties in the reality, and requires post-processing to find a motion trajectory. Based on the passive reconfigurability of underactuated hands [12]–[14], we in this work show that a common type of pre-grasp manipulation, top-sliding, can be planned by integrating motion and grasp planning and it operates directly on the geometrical information. Therefore, it can work on novel objects and does not rely on complicated physics models for planning and control.

III. Problem Formalization and Preliminaries

In this section, we formally define the top-sliding manipulation problem and introduce necessary preliminaries to address this problem. As a major component of our system, we develop a sampleable region based constrained motion planning algorithm. Thereafter, the top-sliding manipulation planning for grasping thin objects will be formulated as an integrated motion and grasp planning problem in Section IV.

A. Top-sliding Manipulation

Top-sliding manipulation is a type of nonprehensile interaction between the robot and object to reconfigure the object in $SE(2)$. For this, the robot is required to make contacts at the top of the object and keep the contacts stabilized during the entire manipulation, so that the object can translate and rotate with the robot's end-effector in $SE(2)$. Depending on the robot used, the contacts can be made by single or multiple fingertips, or by the hand palm.

Let $\mathcal{C} \subset \mathbb{R}^{(d_a+d_h)}$ be the robot's configuration space, and accordingly $\mathcal{C}_{free} \subset \mathcal{C}$ be the collision-free subspace. Formally, represented by $\mathcal{S}_{top} \subset \mathbb{R}^3$ the top surface of an object in the world frame, any configuration $\phi \in \mathcal{C}_{free}$ during top-sliding manipulation should satisfy $\Gamma(\phi) \in \mathcal{S}_{top} \times SO(3)$ to ensure stable contacts on $\mathcal{S}_{top}$ with relative orientation from $SO(3)$, where $\Gamma : \mathcal{C} \rightarrow SE(3)$ calculates the forward kinematics for the desired contact area on the end-effector. To make this condition valid, the collision between the desired contact area on the robot end-effector and $\mathcal{S}_{top}$ is neglected.

Assuming a pair of start and goal configurations, $\phi_s, \phi_g \in \mathcal{C}_{free}$, is given for a top-sliding manipulation task. Using stable contacts between the robot's end-effector and $\mathcal{S}_{top}$, we need to find a continuous path τ to move the object from its initial pose to the desired goal pose. Concretely, the aim is to find a continuous trajectory $\tau \in \Xi^* \subset \Xi : [0,1] \rightarrow \mathcal{C}_{free}$, such that:

$$\begin{aligned} &\tau(0) = \phi_s \wedge \tau(1) = \phi_g \\ &\Gamma(\phi_s) \in \mathcal{S}_{top} \times SO(3) \\ &T_w^o(t)\Gamma(\phi_t) \equiv T_w^o(0)\Gamma(\phi_s), \quad \forall t \in [0,1] \end{aligned} \tag{1}$$

where Ξ is a space of trajectories and Ξ^* is a subspace ensuring that the end-effector always keeps a fixed contact with the object throughout the path. $T_w^o(t) \in SE(3)$ is the transformation from

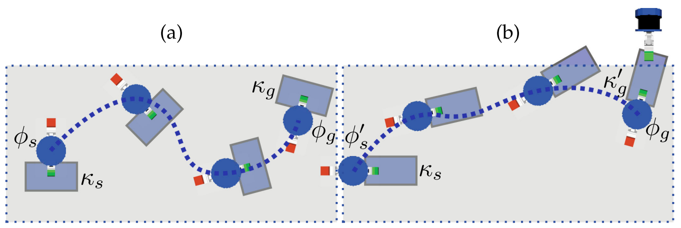

Fig. 2. For the same initial object pose κ_s and final hand pose ϕ_g, top-sliding manipulation can slide the object (blue) to different final poses κ_g and κ'_g dependent on the initial hand poses ϕ_s and ϕ'_s. (a) The object was slid to an ungraspable pose. (b) The object can be grasped from the support surface's side (gray) after sliding.

the world frame to the object frame at time t. The last condition in Eq. (1) enforces that the end-effector keeps the same contact pose on $\mathcal{S}_{top}$ throughout the sliding motion. This condition prevents the end-effector's contact to translate or rotate on the object surface and removes the physical uncertainty involved in the sliding process.

By definition, we can see that the object's movement during sliding is completely controlled by the motion of the end-effector. However, it should be noticed that the object's final pose is not uniquely determined by the goal configuration ϕ_g of the robot. As the main goal of the manipulation, we need to derive how the object's pose changes in terms of the hand movement. Denoted by $\kappa \in SE(3)$ the object's pose in the world frame, an object's final pose κ_g of a top-sliding process is also determined by the start configuration ϕ_s of the robot. Concretely, let inv$(\cdot)$ be the inverse of a transformation in $SE(3)$, the relative pose between the object and the end-effector during the sliding process is constrained by Eq. (2) and depicted in Fig. 2.

$$\kappa_t = \Gamma(\phi_t) \cdot \text{inv}(\Gamma(\phi_s)) \cdot \kappa_s, \quad \forall t \in [0, 1] \tag{2}$$

where κ_t becomes the final pose of the object when $t = 1$ and $\phi_t = \phi_g$. Considering both Eq. (1) and Eq. (2), given an object start pose κ_s, the problem of manipulation planning finally boils down to the problem of finding a pair of (ϕ_s, ϕ_g) and a $\tau \in \Xi^*$ to connect them, such that the object will be moved to $\kappa_g \in \mathcal{G}$, with $\mathcal{G} \subset SE(3)$ denoting the goal region within which the object can be grasped.

To this end, this problem can be addressed from two alternative perspectives. In one way, we can first pre-compute start and goal poses (κ_s, κ_g) for the object, and then find the corresponding pair of robot configurations (ϕ_s, ϕ_g), for which we compute a path τ in the robot configuration space to connect. More efficiently, as proposed in this letter, we actively sample a number of ϕ_s and ϕ_g from some valid regions during planning. While randomly connecting the sampled configurations, we verify each connected pair of (ϕ_s, ϕ_g) in terms of the corresponding (κ_s, κ_g), which can be calculated by Eq. (2), to decide what pair of (ϕ_s, ϕ_g) is a valid solution.

B. Constrained Motion Planning

Object manipulation in general can pose certain constraints on the robot configurations, such as a pose constraint to keep the end-effector always in contact with the object. Most of such constraints form manifolds that occupy extremely small volumes in $\mathcal{C}_{free}$. Planning a trajectory in such manifolds using pure sampling-based methods is therefore unlikely to succeed. To enable efficient motion planning for top-sliding manipulation, we develop a motion planner based on the *Constrained Bi-directional Rapidly-Exploring Random Tree (CBiRRT)* [16]. Given an explicit pair of start and goal configurations, the original CBiRRT generates a trajectory to connect them by randomly sampling in the robot configuration space, and projects the constraint-violating samples onto the manifolds by locally exploring the constraint gradients.

As defined in Section III-A, top-sliding manipulation planning requires to find a pair of (ϕ_s, ϕ_g). However, we in this work do not compute such pairs explicitly prior to motion planning, since it is likely that some arbitrary pairs are difficult to be connected, or even do not have motion solutions, resulting in the planner having to restart with a different pair. Doing so makes the planning much less efficient since the planner needs to always take some time to figure out there is no solution until it restarts. Therefore, we extend the CBiRRT planner to work with sampleable regions so that the planner can work without requiring explicit start and goal configurations as the input. Differently from CBiRRT2 [26], which samples multiple roots for goal configurations and project them to a constraint manifold, our planner samples roots for both start and goal configurations directly from valid regions. However, note that in our planner the configurations sampled from the goal region are not guaranteed to result in a valid motion solution, our planner needs to verify each connected start and goal pair to ensure the validity of the connection.

Concretely, we denote $\mathcal{C}_s, \mathcal{C}_g \subset \mathcal{C}_{free}$ as the start and goal regions which the planner can sample from. Rather than directly growing a forward tree and a backward tree to connect a pair of known $(\phi_s, \phi_g) \in \mathcal{C}_s \times \mathcal{C}_g$, the planner needs to sample $\phi_s^i \in \mathcal{C}_s$ and $\phi_g^j \in \mathcal{C}_g$ to construct a start forest $\Pi_s = \{T_s^i | i = 1, \ldots, n\}$ with T_s^i rooted at ϕ_s^i, and a goal forest $\Pi_g = \{T_g^j | j = 1, \ldots, m\}$ with T_g^j rooted at ϕ_g^j. While the start and goal roots are being sampled, the planner in the meantime grows the forests bidirectionally to connect the start and goal configurations. The extended CBiRRT with sampleable regions is summarized in Algorithm 1, where Π_a and Π_b denote the swappable forests that switch between start and goal forests.

For each tree extension step, NearestTree$(\cdot)$ returns the nearest node in the whole forest as well as the corresponding tree to be extended from. The function ConstrainedExtend$(\cdot)$ from the classical CBiRRT is an unidirectional tree extension procedure that tries to connect a configuration in the tree towards a given target configuration [16]. If an intermediate configuration violates the given constraints, the function will try to project the configuration to a constraint manifold to ensure the validity of the connection, and will terminate if the projection failed or the target is reached.

We can see in Algorithm 1 that the algorithm takes only two sampleable regions as inputs and the forests are automatically constructed. A new tree root is added into the forest in each iteration as long as the CanAddRoot$(\cdot)$ function allows. When a random configuration is sampled, the planner needs to find not only the nearest configuration, but also the corresponding tree, from which ConstrainedExtend$(\cdot)$ will extend towards the sampled configuration. Once a connection is found, the planner triggers the IsPairValid$(\cdot)$ function to check whether the

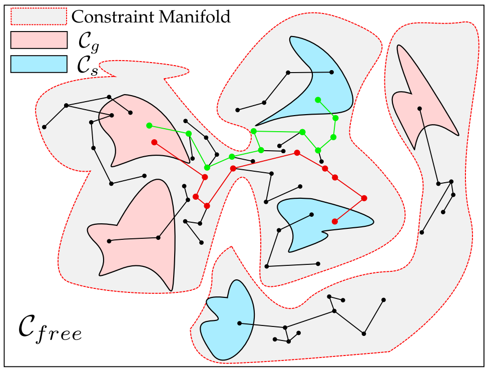

Fig. 3. A schematic plot of Algorithm 1. Start and goal configurations are being sampled and added to the respective forests as roots for different trees. Once a pair of start and goal is connected, it can be verified as invalid (red path), as required by the task, by the IsPairValid(·) function. A path is considered a solution only if it makes a valid connection (green path).

Algorithm 1: CBiRRT with Sampleable Regions.

```
Input: $\mathcal{C}_s, \mathcal{C}_g$
Output: $\phi_s, \phi_g, \tau$
 1: $\Pi_a$. Init($\mathcal{C}_s$), $\Pi_b$. Init($\mathcal{C}_g$)          ▷ Initialization
 2: while Time. Available() do                           ▷ Main Loop
 3:    if CanAddRoot($\Pi_a, \Pi_b$) then
 4:       $\Pi_a$. AddFromRegion()                      ▷ Add Random Root
 5:    end if
 6:    $\phi_{rand}$ ← RandomConfig()
 7:    $(T_a^{near}, \phi_a^{near})$ ← $\Pi_a$. NearestTree($\phi_{rand}$)
 8:    $\phi_a^*$ ← $T_a^{near}$. ConstrainedExtend($\phi_{rand}, \phi_a^{near}$)
 9:    $(T_b^{near}, \phi_b^{near})$ ← $\Pi_b$. NearestTree($\phi_a^*$)
10:    $\phi_b^*$ ← $T_b^{near}$. ConstrainedExtend($\phi_a^*, \phi_b^{near}$)
11:    if $\phi_a^* = \phi_b^*$ then                      ▷ A Connection Found
12:       $(\phi_s, \phi_g)$ ← GetRoots($T_a^{near}, T_b^{near}$)
13:       if IsPairValid($\phi_s, \phi_g$) then            ▷ Start-Goal
                                                          Validation
14:          $\tau$ ← ExtractPath($T_a^{near}, T_b^{near}, \phi_a^*$)
15:          return $\phi_s, \phi_g, \tau$,
16:       end if
17:    end if
18:    Swap($\Pi_a, \Pi_b$)                              ▷ Swap Direction
19: end while
```

connected pair satisfies certain criteria, before outputting the connection path as a solution.

A schematic plot of Algorithm 1 is depicted in Fig. 3, we can see that it is possible that a connected pair of start and goal is invalid for the task. It is also possible that starts and goals are isolated in disconnected constraint manifolds and cannot be connected via a single motion. As will be detailed in Section IV, by designing certain functions in a problem specific manner, solving a constrained motion planning problem using Algorithm 1 will address the top-sliding manipulation problem defined in Section III-A.

IV. Integrating Grasp and Motion Planning

Next, we describe how the support surface and the object are geometrically represented, as well as defining the sampleable

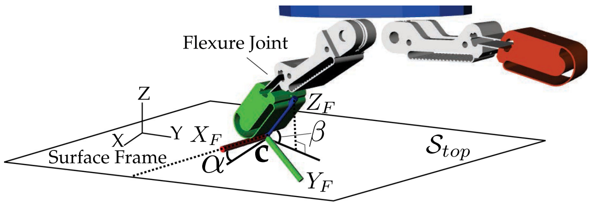

Fig. 4. A fingertip is in contact with the top surface of an object at $\mathbf{c} \in \mathcal{P}_o$. The X_F-axis (red) of the fingertip frame is always aligned in the X-Y plane of $\mathcal{S}_{top}$. The angle α represents the yaw rotation of X_F-axis about the surface normal. β is the pitch angle between the Z_F-axis (blue) of the fingertip frame and the object surface.

regions for both start and goal configurations. Thereafter, we formulate top-sliding manipulation as an integrated grasp and motion planning problem and solve it using Algorithm 1.

A. Object Geometry and Start Region

In order to handle arbitrary object geometries and not rely on any shape parameterizations, we represent the object's top surface $\mathcal{S}_{top}$ as a point cloud $\mathcal{P}_o = \{p_i \in \mathcal{S}_{top} \mid i = 1, \ldots, n_o\}$. As we focus on thin planar objects in this work, we can assume that $\mathcal{P}_o$ can uniquely represent the object's geometry, since the other side of the object is merely mirrored.

We define a contact $\mathbf{c} \in \mathcal{P}_o$ as shown in Fig. 4. We can see that the X_F-axis of the fingertip frame is aligned in the X-Y plane of the surface frame to make a line contact, which can exert both translational and torsional forces at the contact to ensure that the object is translated and reoriented completely in terms of the fingertip's movement, as required in Eq. (2). To describe the fingertip's pose, we denote by α the yaw angle of fingertip about the surface normal, and by β, the pitch angle between the fingertip and the surface plane. For other fingertip models, such as area-contact fingertips, the same notations can be used as long as Eq. (2) is satisfied.

Considering the second constraint in Eq. (1), we need to ensure that a start configuration of top-sliding has the end-effector in contact with $\mathcal{S}_{top}$. Denoted by $r(\alpha, \beta) \in SO(3)$ the fingertip's orientation when it is in contact with the object surface, we define a sampleable region $\mathcal{C}_s \subset \mathcal{C}_{free}$ for start configurations as:

$$\mathcal{C}_s = \{\widetilde{\Gamma}(\omega) | \omega \in \Omega_s\}$$
$$\Omega_s = \mathcal{P}_o \times \{r(\alpha, \beta) | \alpha \in [0, 2\pi), \beta = \beta_0\} \tag{3}$$

where β_0 is a user-defined pitch angle and $\widetilde{\Gamma} : SE(3) \rightarrow \mathcal{C}$ calculates the inverse kinematics of the robot given a fingertip's pose ω. Therefore, we can see that the sampleable region is an infinite set. This set contains robot configurations that can make contacts at any point in $\mathcal{P}_o$ with a pitch angle β_0 and an unconstrained yaw angle α. Sampling from $\mathcal{C}_s$ guarantees that the robot will always start with a valid configuration to slide the object.

B. Support Surface Geometry and Goal Region

In this work, a planar support surface $\mathcal{S}_{surf} \subset \mathbb{R}^3$, on which the object is initially located and then manipulated, is modeled as a polygon and represented by a list of ordered edge points

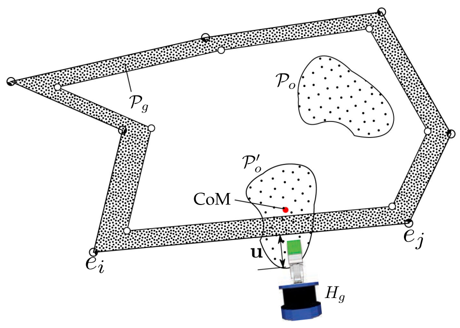

Fig. 5. A support surface is represented as a polygon with edge points $\mathcal{Q} = (e_1, \ldots, e_{n_s})$. $\mathcal{P}_g$ is a set of points near the surface edges. An object represented by point cloud $\mathcal{P}_o$ is slid to $\mathcal{P}'_o$ located near one of the surface edges. u is the largest distance between the edge (e_i, e_j) and object's points which stick out of support surface. The object can be grasped using the hand pose H_g if u is large enough.

$\mathcal{Q} = (e_1, \ldots, e_{n_s}),\ e_i \in \mathcal{S}_{surf}$. As shown in Fig. 5, when the object is located near an edge (e_i, e_j), some of the object is sticking out of the support surface to allow for a grasp from the side.

For pre-grasp top-sliding, it is intuitive that the robot's goal configurations should also have the fingertip poses around the surface edges, so that the fingertip can potentially move the object to a graspable pose. Therefore, using the same notations α and β to describe the yaw and pitch angles between the fingertip and the support surface, we define a sampleable region $\mathcal{C}_g \subset \mathcal{C}_{free}$ for goal configurations as:

$$\begin{aligned} \mathcal{C}_g &= \{\widetilde{\Gamma}(\omega) | \omega \in \Omega_g\} \\ \Omega_g &= \mathcal{P}_g \times \{r(\alpha, \beta) | \alpha \in [0, 2\pi), \beta = \beta_0\} \end{aligned} \tag{4}$$

where $\mathcal{P}_g \subset \mathbb{R}^3$ is a set of points near the surface edges. As depicted in Fig. 5, $\mathcal{P}_g$ is obtained by contracting the surface edge points towards the inside of the surface, so that an area is formed between the original surface and the contracted surface. Note that the points in $\mathcal{P}_g$ have a different height than the support surface. To ensure the constraint in Eq. (2), the height of $\mathcal{P}_g$ is lifted by the object's thickness.

Sampling goal configurations from $\mathcal{C}_g$ will ensure that the fingertip will be in contact with the object $\mathcal{S}_{top}$, and that the object is near the table edges. However, this does not guarantee that the object is in a graspable pose. Next, we will assemble the developed components together to address the pre-grasp top-sliding problem, so that the object will be slid from its original pose to some graspable pose near the edges of the support surface.

C. Integrated Planner

Considering a thin object represented by point cloud $\mathcal{P}_o$ on a support surface represented by $\mathcal{Q}$, the object is graspable from the side of the support surface only if the object's pose satisfies three conditions: 1) The object has enough portion sticking out of the surface for the hand to grasp; 2) The object's center of mass (CoM) is inside the surface $\mathcal{Q}$ so that the object can stay on the surface before grasping; and 3) There exists a motion solution to move the hand to the grasping pose. Using the sampleable regions for start and goal configurations, we are now able to address this problem as an integrated grasp and motion planning problem and solve it using Algorithm 1.

1) Pair Validation: Formally, the integrated planning is conducted by the combination of sampleable regions and the IsPairValid(·) function. As described in Section IV-A, the roots of the start forest are sampled from $\mathcal{C}_s$ to ensure fingertip contacts on the object $\mathcal{P}_o$, while the roots of the goal forest are sampled from $\mathcal{C}_g$ to finally relocate the object near the support surface's edges. Once a pair of start and goal is connected by a path $\tau \in \Xi^*$ using the motion planning procedure in Algorithm 1, the function IsPairValid(·) will be invoked to verify whether the object can be moved to a graspable pose by τ.

Given a goal configuration ϕ_g, we can compute the object's final pose using Eq. (2) dependent on which start configuration ϕ_s is paired. As such, we can transform $\mathcal{P}_o$, which is a point cloud representing the object geometry in its original pose, to its final pose $\mathcal{P}'_o$, as depicted in Fig. 5. Thereafter, we check for each $p_i \in \mathcal{P}'_o$ whether it is within the surface defined by the polygon $\mathcal{Q}$ using the point-in-polygon algorithm [27]. We then obtain the partial point cloud $\mathcal{P}_o^{out} \subset \mathcal{P}'_o$ which contains all the points sticking out of the support surface $\mathcal{Q}$. As shown in Fig. 5, we calculate the largest distance, denoted by $u \in \mathbb{R}^+$, between $\mathcal{P}_o^{out}$ and the edge (e_i, e_j), which is the closest edge to the center of mass of $\mathcal{P}_o^{out}$.

Lastly, denoting u^* as the minimum contact size at the fingertip, if $u \geq u^*$ and the CoM of $\mathcal{P}'_O$ is inside the surface polygon $\mathcal{Q}$, we need to verify whether $\mathcal{P}_o^{out}$ is kinematically reachable for grasping. For this, as shown in Fig. 5, we compute a hand pose $H_g \in SE(3)$ to reach the object perpendicularly to the edge in the $\mathcal{Q}$ plane and grasp at the the geometric center of $\mathcal{P}_o^{out}$. Finally, if H_g can be reached with a motion starting from ϕ_g, IsPairValid(·) will return True and the planner will output the pair (ϕ_s, ϕ_g) and τ as the solution.

2) Root Sampling: As described in Algorithm 1, root sampling is controlled by the function CanAddRoot(·). Dependent on the planning status, we wish to progressively control whether a new root should be added to the forests. In case there are not enough roots in each forest, the planner will struggle to connect the limited number of pairs, which could be difficult or impossible to connect as depicted in Fig. 3. In another case, if there are too many roots, the planner will be slowed down as it is distracted by many trees.

As such, denoted by $N_s, N_g \in \mathbb{N}^+$ the number of roots in the start and goal forests Π_s, Π_g, the planner decides whether to add new roots in the forests by:

$$\text{CanAddRoot}(\Pi_s, \Pi_g) = \begin{cases} \text{True}, & e^{-\frac{N_s + N_g}{N^*}} > rand(0,1) \\ \text{False}, & \text{Otherwise} \end{cases} \tag{5}$$

where $N^* \in \mathbb{N}^+$ is a number indicating how conservative the planner behaves in adding new roots. Larger number makes it less conservative. We can see that the probability of adding new roots is never zero. This guarantees the probabilistic completeness that, given infinite amount of time, the planner will finally find a solution as long as there is such a pair resulting in a successful top-sliding manipulation.

3) Constraints and Relaxations: Since the fingertip is always in contact with the object's top surface during sliding manipulation, in addition to collision checking, the

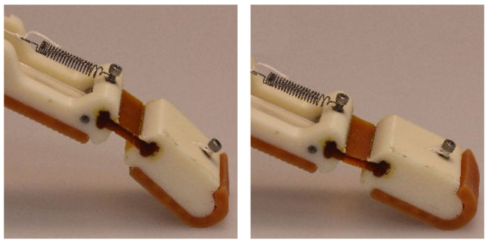
Fig. 6. Passive reconfigurability of the flexure joint on the Yale T42 hand. *Left:* The finger in its resting configuration. *Right:* The finger is passively reconfigured by the contact force at the fingertip.

ConstrainedExtend($\cdot$) function in Algorithm 1 needs to take an extra condition to ensure that the fingertip never moves out of the support surface. This is strictly enforced to avoid object dropping from the support surface during sliding, and can be implemented based on the forward kinematics $\Gamma(\cdot)$ and the point-in-polygon method.

Recall that the last constraint in Eq. (1) enforces that the relative pose between the fingertip and the object is fixed. This forms a constraint manifold occupying an infinitesimal volume in $\mathcal{C}_{free}$ and renders the planning very inefficient, since almost all the random samples would require to be projected to the exact constraint manifold. Similar to [16], we relax the constraints to expand the manifold with a small neighborhood to accelerate the planning. As depicted in Fig. 6, this relaxation is possible since the underactuated finger provides passive reconfigurability to compliantly adapt the hand configuration when external forces are applied at the fingertip. To this end, based on the motion constraint in Eq. (2), and the contact constraints in sampleable regions in Eq. (3) and Eq. (4), we define two constraint relaxations for the contact pitch angle $\Delta_\beta \in \mathbb{R}$ and for the contact depth $\Delta_Z = [\Delta_Z^-, \Delta_Z^+] \in \mathbb{R}^2$, $\Delta_Z^- \leq \Delta_Z^+$. The difference, $\Delta_z^* = \Delta_Z^+ - \Delta_Z^-$, between the relaxation bounds on contact depth is termed as *Reconfiguration Range*, which essentially determines the volume of the expanded constraint manifold.

Intuitively, these relaxations are feasible since both of them do not affect the object's pose constrained on a support surface in $SE(2)$. The relaxation Δ_β enables the pitch angle to vary in a symmetric range $[\beta_0 - \Delta_\beta, \beta_0 + \Delta_\beta]$ to allow reasonable contact rolling. Δ_Z enables the contact to virtually penetrate the object surface and vary its depth in an asymmetric range $[z_s - \Delta_Z^+, z_s - \Delta_Z^-]$, where $z_s \in \mathbb{R}$ is the height of the object surface $\mathcal{S}_{top}$. We can see that the contact is kept below the height of $\mathcal{S}_{top}$ by at least Δ_Z^-, in order to keep a minimum amount of finger reconfiguration at the contacting fingertip to exert enough force to slide the object. Furthermore, Δ_Z^+ allows more additional finger reconfiguration for the path planning to improve the efficiency. Based on the relaxed constraints, we conduct the manifold projection by employing a gradient-based method for the ConstrainedExtend($\cdot$) function as proposed in [16].

V. Experiments

We evaluate our approach from 4 perspectives: 1) We show that the proposed approach can work with arbitrary objects and support surfaces, and that the planner is not sensitive to object scales; 2) We quantitatively evaluate how the planner's efficiency can be improved by the reconfiguration range Δ_Z^*; 3) We compare our method with two baseline methods; and in the end, 4) We analyze how the planner's performance is affected by its conservativeness in adding more start and goal configurations during planning, as defined by the factor N^* in Eq. (5).

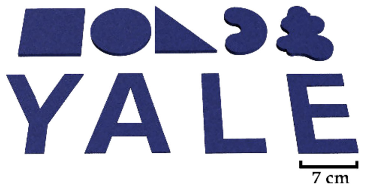

Fig. 7. Plastic test objects used in the experiments: Square, Circle, Triangle, Crescent and Irregular, as well as the "Y", "A", "L", "E" letters.

I the experiments, we used 9 plastic thin flat objects shown in Fig. 7. The planner has been implemented for a WAM arm installed with a Yale T42 hand in both Gazebo simulator [28] and on a real platform. The implementation is partially based on the OMPL library [29], [30] and written in Python. The reported evaluations were run on a machine with an Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz $\times$ 4 and a 8GB RAM running Ubuntu 16.04. Some example real robot experiments can be seen in Fig. 1 and Fig. 8. Since the contact pitch relaxation Δ_β can be applied to both fully-actuated hands and underactuted hands, we fix it to be $\Delta_\beta = 0.1rad$ in our evaluation and focus on the reconfiguration range Δ_Z. As will be shown in Section V-7, we set $N^* = 100$ for better performance unless elsewhere stated.

4) Geometries of Objects and Support Surfaces: The planner has been evaluated using 5 objects on 4 different support surfaces. As illustrated in Fig. 9, our planner is able to plan pre-grasp top-sliding manipulations for the robot to slide different objects on different support surfaces to graspable poses. As described in Section IV-B, all the support surfaces are represented as polygons, and for the round table, we used 100 edge points to approximate the shape of the circle.

We randomly sampled 50 initial object poses on each of the 4 support surfaces and applied the planner on 5 test objects to conduct in total $50 \times 4 \times 5 = 1000$ experiments in simulation. In addition, we scaled down the objects by half and repeated same experiments. From the results summarized in Fig. 10, we can see that for different object geometries and sizes, the average planning time is similar. Moreover, the planner performed equally well for 4 different support surfaces. These results imply that the geometrical representations of objects and support surfaces developed in Section IV are able to work with any shapes and are insensitive to geometries and scales.

5) Reconfiguration Range: We now evaluate the impact of Δ_Z, which is only available for underactuted hands. According to the mechanical properties of the employed T42 hand, we set $\Delta_Z^- = 0.5$ cm to ensure enough contact force, and vary the reconfiguration range Δ_Z^* for evaluation.

For this experiment, we used the square table and randomly sampled 100 initial object poses for the 5 objects and applied the planner with 3 different Δ_Z^* values to conduct in total 1500 runs. As reported in Fig. 11, with a reconfiguration range of $\Delta_Z^* = 2$ cm, the planner was able to achieve a success rate of higher than 90% within 20 seconds and finally 99% at 60

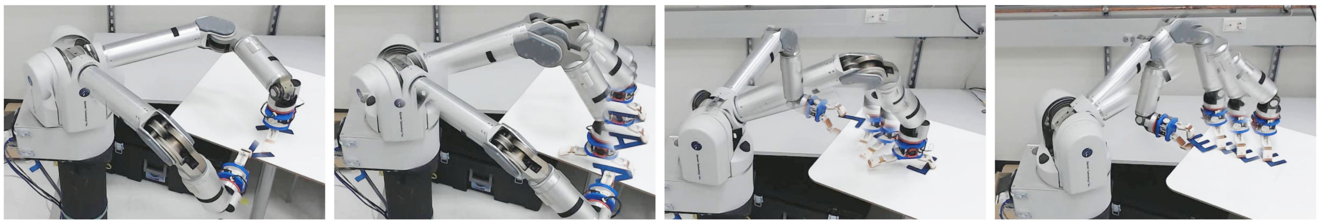
Fig. 8. Real executions of the proposed planner using a WAM arm installed with a Yale T42 hand. The objects are perceived using a Kinect sensor installed over the tabletop. The plastic “Y”, “A”, “L” and “E” letters are grasped after executing the pre-grasp top-sliding manipulations planned by our planner. More example executions can be found in the supplementary video.

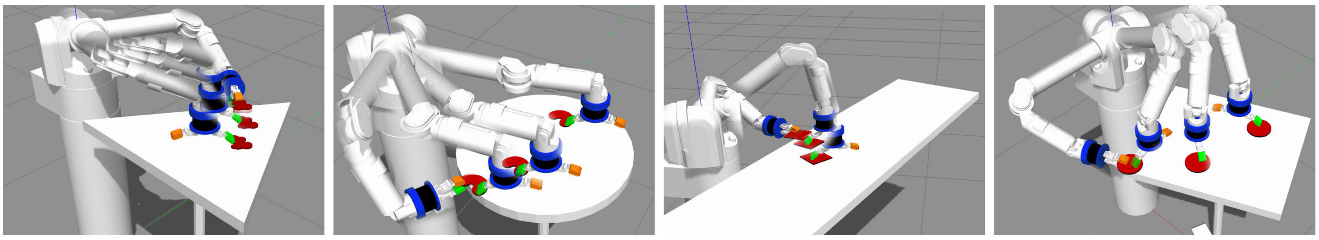
Fig. 9. The proposed approach is able to plan pre-grasp sliding manipulations for arbitrary geometries of support surfaces and target objects. The example support surfaces are: triangle table, round table, long table, and square table.

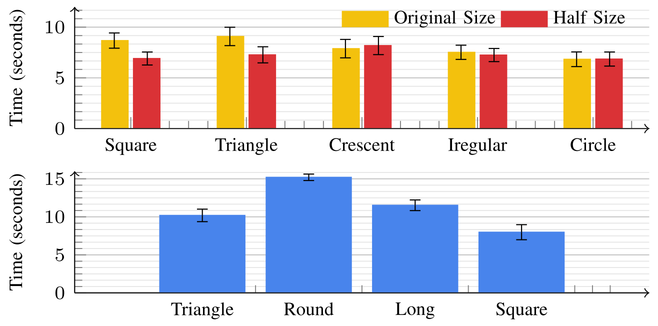

Fig. 10. *Upper*: Average planning time for 5 test object geometries. *Lower*: Planning time averaged over 5 objects on different support surfaces. The planner was executed with constraint relaxation factors $\Delta_\beta = 0.1rad$ and $\Delta_Z = [0.5 \text{ cm}, 2.5 \text{ cm}]$. The error bars show standard errors.

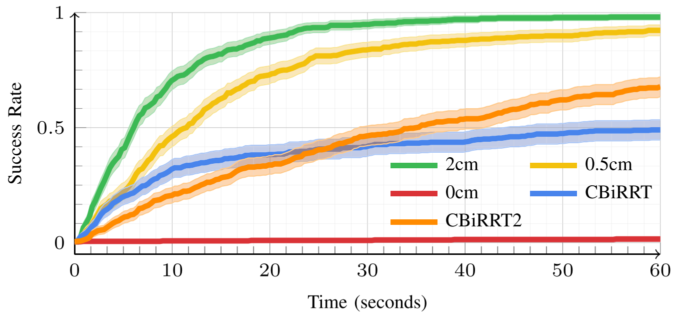

Fig. 11. The planning success rate as a function of planning time, which can be interpreted as a chance of planning success. The shaded areas show the 95% Wilson confidence interval. The reconfiguration range is set to be $\Delta_Z^* = 0$ cm, 0.5 cm and 2 cm. The blue and orange curves show the performance of two baseline methods with $\Delta_Z^* = 2$ cm for comparison.

seconds. However, it took the planner 53 seconds to reach a success rate of 90% when the reconfiguration range was set to $\Delta_Z^* = 0.5$ cm. In the worst case, when we set Δ_Z^* to be 0, the planner was able to successfully generate plans for only 2% of the test cases within the time budget of 60 seconds. This result shows that the passive reconfigurability enabled by the underactuted hand is essential and that it significantly affects the performance.

6) Baseline Comparison: In addition to the proposed approach, we implemented two other methods based on CBiRRT and CBiRRT2 and repeated the experiments as in Section V-5. Briefly, for CBiRRT-based method, we pre-compute valid start and goal robot configuration pairs and only ask the planner to plan the trajectory under motion constraints. While for CBiRRT2-based method, only start robot configuration is pre-computed and the goals are sampled.

As reported in Fig. 11, both baseline planners performed worse than our approach. This is because of two reasons: a) it is possible that there does not exist a path to connect the given start and goal pair for the CBiRRT-based method, and b) a given start configuration can only be connected through some narrow passages in the constraint manifold. It is interesting to notice that the CBiRRT2-based method is better than the CBiRRT-based method only when the time budget is long enough. This is because the CBiRRT2-based method is inherently more expensive since it has to manage a forest and needs to verify solution pairs. However, when given enough time, it will start to benefit from having more goals which could give it easier pairs to connect. As such, the proposed sampleable regions for both starts and goals can enable the planner to perform much better, since it is not forced to focus on any explicit starts or goals. Instead, it tries to connect any pair that is feasible and easy in motion.

7) Root Sampling: As defined in Eq. (5), the parameter N^* controls how conservative the planner behaves in adding more roots. A larger number makes it less conservative and vice versa. To evaluate how this parameter affects the planner’s performance, we repeated the experiments in Section V-5 for 5 values of N^* and report in Fig. 12. As indicated by the results, when the value is small, the planner is too conservative and does not benefit much from having many starts and goals. On the other hand,

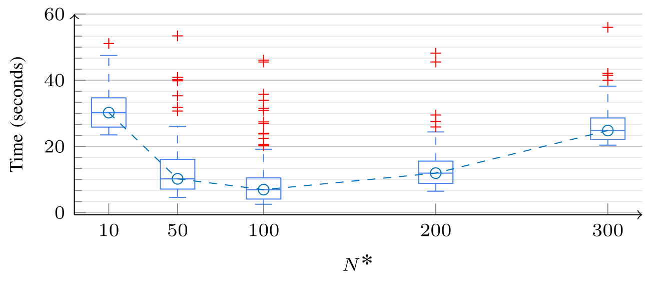

Fig. 12. Average planning time in terms of how conservative the planner adds new start and goal configurations, as determined by N^* in Eq. (5). In this experiment, the reconfiguration range is set to $\Delta_Z^* = 2$ cm.

when the value is too big, the planner will suffer from being distracted by too many roots in the forest and being computationally too expensive. In our evaluations, the best performance was observed when $N^* = 100$.

VI. Conclusion

In this work, we addressed the problem of pre-grasp sliding manipulation planning for grasping thin objects on planar support surfaces. In particular, leveraged on the passive reconfigurability of underactuated hands, we focused on top-sliding manipulation and formulated the problem as an integrated motion and grasp planning problem with constraints. Rather than explicitly pre-computing start and goal configurations for manipulation, and later on connect them using a motion planner in a separate step, we developed configuration sampleable regions to enable our planner to automatically generate start and goal candidates. By extending the classical CBiRRT algorithm with the sampleable regions, our planner constructs start and goal forests and bidirectionally connect them while respecting task constraints. The integrated motion and grasp planning was realized by a superimposed pair validation to ensure the generated motion in the robot's configuration space can slide the object to a graspable pose. By relaxing the task constraints based on the passive reconfigurability of underactuated hands, we showed that the robot can exert appropriate forces at the contacts to achieve manipulation without requiring complicated force planning or control.

The proposed approach was implemented in both simulation and on a real robot. The experiments validated our problem formulation, and the evaluation results showed that our approach outperforms two baseline planners and that the passive reconfigurability of underactuated hands can significantly improve the planning efficiency. In future work, we plan to extend our approach to work in cluttered environments with multiple objects on the support surface, as well as enabling it to work with un-planar surfaces.

References

[1] s J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis – A survey," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 289–309, Apr. 2014.
[2] M. Roa and R. Suárez, "Grasp quality measures: Review and performance," *Auton. Robots*, vol. 38, no. 1, pp. 65–88, 2015.
[3] A. Kimmel, R. Shome, Z. Littlefield, and K. E. Bekris, "Fast, anytime motion planning for prehensile manipulation in clutter," 2018, arXiv preprint arXiv:1806.07465.
[4] J. A. Haustein, K. Hang, and D. Kragic, "Integrating motion and hierarchical fingertip grasp planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3439–3446.
[5] J. Mahler *et al.*, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Robot.: Sci. Syst.*, 2017.
[6] K. Hang *et al.*, "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation," *IEEE Trans. Robot.*, vol. 32, no. 4, pp. 960–972, Aug. 2016.
[7] L. Y. Chang, G. Zeglin, and N. S. Pollard, "Preparatory object rotation as a human-inspired grasping strategy," in *Proc. IEEE Int. Conf. Humanoid Robots*, 2008, pp. 527–534.
[8] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Preparatory object reorientation for task-oriented grasping," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2016, pp. 893–899.
[9] D. Kappler, L. Y. Chang, N. S. Pollard, T. Asfour, and R. Dillmann, "Templates for pre-grasp sliding interactions," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 411–423, 2012.
[10] J. E. King *et al.*, "Pregrasp manipulation as trajectory optimization," in *Proc. Robot.: Sci. Syst.*, 2013.
[11] J. Zhou, J. A. Bagnell, and M. T. Mason, "A fast stochastic contact model for planar pushing and grasping: Theory and experimental validation," 2017, arXiv preprint arXiv:1705.10664.
[12] C. D. Santina, G. Grioli, M. Catalano, A. Brando, and A. Bicchi, "Dexterity augmentation on a synergistic hand: The pisa/iit softhand+," in *Proc. IEEE Int. Conf. Humanoid Robots*, 2015, pp. 497–503.
[13] M. Ciocarlie *et al.*, "The velo gripper: A versatile single-actuator design for enveloping, parallel and fingertip grasps," *Int. J. Robot. Res.*, vol. 33, no. 5, pp. 753–767, 2014.
[14] A. M. Dollar and R. D. Howe, "The highly adaptive sdm hand: Design and performance evaluation," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 585–597, 2010.
[15] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *Int. J. Robot. Res.*, vol. 35, nos. 1–3, pp. 161–185, 2016.
[16] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 625–632.
[17] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. IEEE Int. Conf. Robot. Automat.,*, 1992, pp. 2290–2295.
[18] K. Hang, J. A. Stork, N. S. Pollard, and D. Kragic, "A framework for optimal grasp contact planning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 704–711, Apr. 2017.
[19] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *Int. J. Robot. Res.*, vol. 28, no. 7, pp. 851–867, 2009.
[20] K. Tahara, S. Arimoto, and M. Yoshida, "Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 4322–4327.
[21] Y. Zheng, "Computing the best grasp in a discrete point set with wrench-oriented grasp quality measures," *Auton. Robots*, Jul. 2018, doi: 10.1007/s1051401897884.
[22] K. Hang, J. A. Stork, and D. Kragic, "Hierarchical fingertip space for multi-fingered precision grasping," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2014, pp. 1641–1648.
[23] M. R. Dogar and S. S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2010, pp. 2123–2130.
[24] M. uddin, M. Moll, L. E. Kavraki, and J. Rosell, "Randomized physics-based motion planning for grasping in cluttered and uncertain environments," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 712–719, Apr. 2018.
[25] K. M. Lynch, "Toppling manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1999, vol. 4, pp. 2551–2557.
[26] D. Berenson, J. E. Chestnutt, S. S. Srinivasa, J. J. Kuffner, and S. Kagami, "Pose-constrained whole-body planning using task space region chains," in *Proc. IEEE Int. Conf. Humanoid Robots*, 2009, pp. 181–187.
[27] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A characterization of ten hidden-surface algorithms," *ACM Comput. Surv.*, vol. 6, no. 1, pp. 1–55, Mar. 1974.
[28] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, Sep. 2004, vol. 3, pp. 2149–2154.
[29] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.
[30] Z. Kingston, M. Moll, and L. E. Kavraki, "Decoupling constraints from sampling-based planners," in *Proc. Int. Symp. Robot. Res.*, 2017.